Selbstgebauter, kompakter, Strom sparender, Mehrkanal-Datenlogger mit PICs

Wettbewerb "Jugend Forscht" 2008

Lucas Jürgens (12 Jahre)

Arbeitsgemeinschaft "Jugend Forscht"
des Christian-Gymnasiums Hermannsburg
Leitung: StD Thomas Biedermann

Inhaltsverzeichnis

Ziel

Mein Ziel ist es einen Datenlogger zu bauen, der möglichst klein und stromsparend ist.

Man soll ihn über einen längeren Zeitraum unbeaufsichtigt und auch im Freien einsetzen können, dabei soll er mehrere Messwerte (z.B. Temperatur, Luftfeuchtigkeit, usw.) gleichzeitig aufnehmen können.

Was ist ein Datenlogger?

Ein Datenlogger nimmt in regelmäßigen Abständen Messdaten auf, die von Sensoren als elektrische Größe bereit gestellt werden, diese können analog oder auch digital sein.

Die gemessenen Daten werden gespeichert und können anschließend per PC ausgewertet werden.

So können z.B. Wetterdaten über einen längeren Zeitraum unbeaufsichtigt aufgenommen werden.

Umsetzung

Damit der Datenlogger kompakt bleibt, verwende ich um die Messdaten aufzuzeichnen einen Mikrocontroller vom Typ PIC (siehe s.3). Ich programmiere den PIC so, dass er Daten wie Messgenauigkeit, Messintervalle und Meßwertanzahl so aufnimmt wie ich es möchte.

So kann ich unterschiedliche Genauigkeit, Zeitintervalle und Meßwerte zwischen den Messungen einstellen.

Anschließend kommen alle Komponenten (der PIC, die Sensoren, Platine, Batterie und ggf. Solarzelle) in ein wetterfestes Gehäuse. So ist der Datenlogger auch draussen in der freien Natur einsetzbar.

Was ist ein PIC?

Ein PIC ist ein kleiner Mikrocontroller, den man über einen PC programmieren kann. Sie werden oft in z.B. Digitaluhren, Handys usw. verwendet. Sie können Tastendrücke und vieles mehr verarbeiteten, so kann man z.B. bei Digitaluhren die Uhrzeit einstellen. Es gibt viele Anbieter, ich verwende in meinem Projekt Mikrocontroller von der Firma Microchip.

Alle PICs verfügen neben dem Prozessor über 2 verschiedene Speicher (Flash und EEPROM), Digital-Ports sowie eine Programmierschnittstelle. Je nach Ausführung können weitere Komponenten vorhanden sein, z.B. AD-Wandler, Timer usw.

Für meinen Datenlogger benutze ich einen PIC mit der Bezeichnung PIC18F452.

Er verfügt über

- 8 A/D Wandler mit 10 Bit Auflösung
- 4 Timer
- 32 KByte Flashspeicher
- 256 Byte EEPROM
- USART-Schnittstelle

Abb. 1: PIC im TQFP-Gehäuse mit 80 Pins in Originalgröße

In den Flashspeicher kommt das Programm, im EEPROM Speicher werde ich die Messdaten speichern.

Die A/D-Wandler (Analog-Digital-Wandler) wandeln die analogen Werte (die Signale von den Sensoren) in binär speicherbare digitale Werte um. Da mir 8 A/D-Wandler zu Verfügung stehen, kann ich mehrere Messdaten gleichzeitig aufnehmen.

Die USART-Schnittstelle erlaubt die Datenübertragung vom und zum PC über dessen serielle Schnittstelle.

PICs sind in verschiedenen Gehäuseformen erhältlich: In der länglichen PDIP-Version ist er für übliche Steckfassungen geeignet, benötigt aber auch am meisten Platz, in der QFN-Version kann er in PLCC-Fassungen gesteckt werden, benötigt aber etwas weniger Platz, und am kompaktesten ist die TQFP-Version (siehe auch Abb. 1, für einen PIC18F8720 mit 80 Pins), mit der er auf Platinenoberflächen gelötet werden kann.

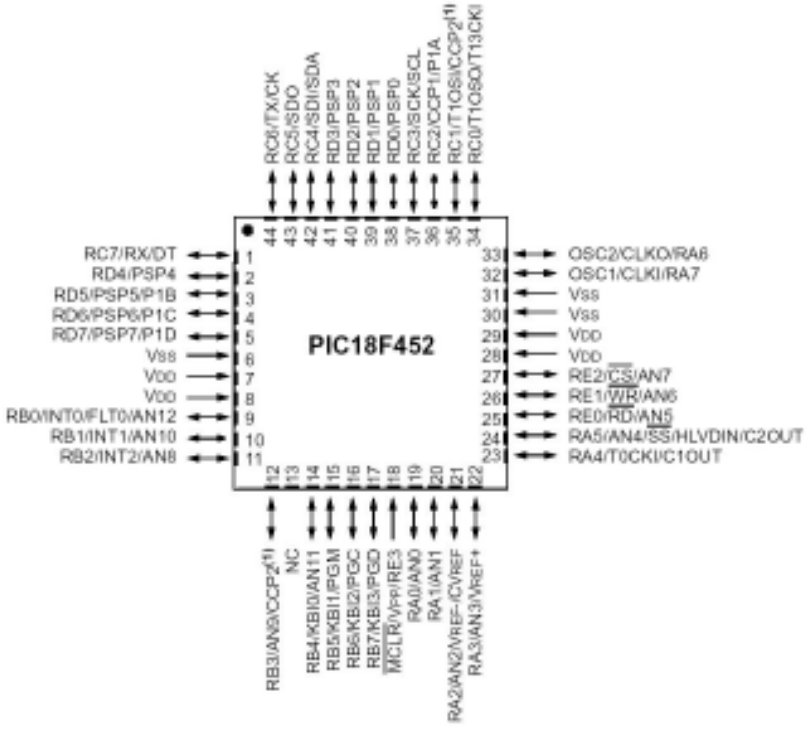

Abb. 2: Pinbelegung des von mir benutzten PIC18F452 im QFN-Gehäuse

Ich benutze die QFN-Version, deren Pinbelegung für meinen PIC Abbildung 2 zeigt.

Ich habe die Microchip-PICs ausgewählt, weil wir für diesen Typ in unserer AG bereits ein passendes Programmiergerät sowie die zughehörige Programmiersoftare haben. Außerdem benötigen sie nur sehr wenige zusätzliche Komponenten und haben eine geringe Stromaufnahme, die man sogar noch weiter reduzieren kann indem man sie langsamer laufen lässt.

Wie programmiert man einen PIC?

Der PIC arbeitet jeden Befehl, der im Programmcode steht, einzeln und nacheinander ab. Mit den geeigneten Befehlen kann ich bewirken, dass z.B. bei einem Tastendruck eine LED blinkt oder ein Messwert aufgenommen und gespeichert wird.

Zuerst schreibt man den Code in einer Programmiersprache, die C sehr ähnlich ist und zusätzlich über bestimmte Befehle verfügt, die für den PIC benötigt werden. Wenn der C-Code fertig ist, wird er kompiliert, das heißt, dass er in eine für den PIC spezielle Maschinensprache umgewandelt wird.

Sowohl für den C-Editor als auch für den Kompiliervorgang benutze ich die kostenlose Studentenversion Mplab von Microchip, die auch die entsprechenden C-Bibliotheken bereitstellt.

Wenn der Compiler keine Fehler ausgibt, wird eine HEX-Datei erstellt, die mit einer speziellen Software unter Verwendung eines einfachen Programmiergerätes auf den PIC geladen werden kann.

Dazu benutze ich das Freeware-Programm IC-Prog. Der PIC wird dazu in den Sokkel des Programmiergeräts gesteckt, was über den LPT Port mit dem Computer verbunden ist. Das Programm löscht den Inhalt des Programmspeichers, überträgt anschließend die Daten an den PIC und überprüft sie, als Ergebnis ist nach einigen Minuten der PIC „gebrannt".

Aufbau eines PIC-Programms in C

Zunächst mssen die Header-Files angegeben werden, auf die der Kompiler zurgreifen muss und die die meisten Befehle einthalten, die für den PIC speziell benötigt werden. Eine entsprechende Zeile lautet z.B.:

```
#inlude <p18f452.h>
```

diese enthält vor allem für diesen PIC die speziellen vordefinierte Bezeichner für die einzelnen Ports sowie die Adressen dieser Ports und besonderer Register.

Anschließend werden die Einstellungen für den Chip vorgenommen, dazu gehören z.B. die Herkunft des Taktsignals, das Verhalten bei zu niedriger Versorgungsspannung usw. Dies geschieht in einem Abschnitt, der beginnt mit

```
#pragma romdata CONFIG
```

und wird abgeschlossen mit der Zeile

```
#pragma romdata
```

Anschließend werden Funktionen deklariert, die in anderen Modulen definiert sind, wie z.B.

```
extern void SleepMS(int MilliSeconds);
```

bzw. die im weiteren Verlauf des Codes benutzt und definiert werden und einen anderen Rückgabewert als die Voreinstellung int haben, z.B.:

```
void BeekSpkr(int freq, int duration);
```

Als nächstes werden die Portregister festgelegt, also ob sie Ein- oder Ausgänge darstellen sollen, z.B.:

```
TRISA = 0b00000000;
```

sowie das Verhalten der einzelnen Interruptquellen eingestellt, z.B.:

```
RCONbits.IPEN = 0;
```

Um eine LED blinken zu lassen sieht der Code dann folgendermaßen aus:

```
while(1)                    // Endlosschleife
{                           // Beginn der Schleife
    PORTA |= 0xff;          // Alle LED's an PortA anschalten.
    SleepMS(1000);          // Warte 1 sek
    PORTA = 0x00;           // Alle LED's wieder ausschalten
    SleepMS(1000);          // Wartet 1 sek
}                           // Ende der Schleife
```

Hinter jeden Befehl kommt ein Semikolon, dann weiß der Compiler, dass der Befehl zu Ende ist.

So könnte man die vier Befehle auch in zwei Zeilen schreiben:

```
PORTA |= 0xff;    SleepMS(1000);
PORTA |= 0x00;    SleepMS(1000);
```

Programmierung des Datenloggers

Für den (noch zu programmierenden) Datenlogger müssen nun nach dem Ausschalten der LEDs die Anlog-Eingangskanäle abgefragt werden. Dazu muss der entsprechende Kanal zunächst aktiviert werden, anschließend muss man mehrere Taktkzyklen warten, bis die Umwandlung des Signals in einen binären Wert abgeschlossen ist. Nun kann dieser Binärwert im statischen RAM (Static RAM) gespeichert werden, das bei meinem PIC bis zu 3840 Byte an Daten aufnehmen kann, dafür benötige ich ein entsprechendes Datenfeld, z.B. mit der Anweisung

```
char data[3800];
```

Das Datenfeld darf nicht den gesamten Speicherplatz belegen, weil ich sonst keinen Platz mehr für weitere benötigte Variablen (z.B. Zähler) habe.

Der letzte Warteschleifenaufruf legt fest, wie lange es dauert, bis die nächste Messung aufgenommen wird.

Statt der Endlosschleife muss ich allerdings nun eine Abbruchbedingung vorsehen, z.B. dass der maximale Zählerstand erreicht ist oder eine Taste gedrückt wurde.

Ein Programm, mit dem ich die Daten auslesen und mit dem USART an den PC übertragen kann, muss ich mir noch überlegen.

Quellen

Abb. 1: Wikipedia - Stichwort „Mikrocontroller"

Abb. 2: Datenblatt der Firma Microchip (PIC18F452.pdf)

MPLAB Download von Microchip.com (C18 Student-Edition)

IC-Prog Download von IC-Prog.com (Version 1.05 C)